Reliability and Redundancy for the Central Services of CDF SAM

Stefan Stonjek

23-Mar-2005

1 Problem

CDF already has many SAM stations installed around the world. This includes SAM stations in Europe, North-America and Asia. To function properly all these SAM station depend on the central SAM services. The main ones are:

- CORBA naming service
- db-server(s)
- central database

As long as these services are up and reachable SAM stations can functions. A failiure in each of the services effects the whole SAM system, but to a different degree.

- A failure in the central database would only effect the db-server. Version 5 db-servers would have needed a manual restart. But version 6 db-server are able to reconnect by themselfs. SAM stations would not be effected by a database failure because they do not talk to the database directly. They would just see very long delays in the db-server answers.
- Tests have shown that current versions of the SAM station are able to deal with a db-server failure. A SAM station would not crach but wait till the db-server is up again.
  Current versions of the SAM client would respond with a reasonable error message to a db-server problem and would point the user in the right direction.
  A db-server problem would prevent stations for starting as would a problem between the db-server and the central database or a problem in the central database..
- A failure in the CORBA naming service (crash or network problem) would crash all SAM stations which were up and connected to that CORBA naming service at that point in time.

Therefore the CORBA nameing service is the most critical single service in the current situation.

2 Approaches

With some effort it would be possible to alter the station and client code to cope with the nonexistance of any nameing service. But this seems to be non-trivial. In addition any naming service and/or db-server downtime would still stop block every SAM station.

A better approach would include redundant nameing services, db-servers and database machines. Since SAM has to write to the central database their is no easy way to achieve redundancy for the database itself.

But it is possible to construct a redundant system of nameing service and db-server, which would prevent SAM station from craching, even during database outages[1]

[1] CDF had to deal with many database outages in recent days, therefore we feel that this is an issue.

3 Setup

The general idea for redudant CORBA nameing services and db-servers consinst of several identical machines which each have a CORBA nameing service and a SAM db-server running.

On each of the machines the db-server would register with the local CORBA nameing service. The name used for this registration would be the same on all machines.

Each SAM station (and SAM client) would have to know about all nameing services. Whenever the station want to talk to a db-server it would contact the nameing service and ask for the db-server. In case a nameing service (or a db-server) is not reachable the station would fail-over to the next CORBA nameing service in the list.

To allow fail-over in case of network problems it might be usefull to have several machines around the world: e.g. two at Fermilab, one in Europe and one in Asia.

4 Test Results

The SAM station is already able to deal with four CORBA nameing services. Theirfore we already configured and tested the system descibed above[2]. We were even able to see a station fail back to another nameing service.

But it was not possible to start a station as long as one of the nameing services the station knows about are unreachable.

In addtion this test showed that the db-server should be close to the database machine itself. The connection between the db-server and the database is more sensitive to delays than the connection between the db-server and a SAM station.

5 Possible Scenarios

There are several ways to allow remote SAM stations to survive a problem at FCC.

- One could change the station and client code which would allow them ride trough a nameing service problem.
- One could set up redundant nameing services as described above.
- One could force every station admin to restart the station whenever FCC has a power problem
- One could redesign the station so that all relevant information are local

Given the fact that CDF already has to server class machines (cdfsam01 and cdfsam02) which could easily configured to act as redundant nameing services and the fact that a SAM station can alreday deal with four nameing services, this setup seems to be the most effective and straigt forward one.

6 Conclusion

It is possible to design a CORBA system without a single-point-of-failure. The database remains the only single-point-of-failure[3].

The SAM station code has to be checked and the startup bug fixed.

The SAM client code needs the ability to fail-over to redundant nameing services.

Remote CDF SAM stations really need the ability to survive FCC problems. For their good we should implement this feature.

[2] We used two nameing services and db-servers at Fermilab: cdfsam01 and cdfsam02

[3] Oracle 10g might offer a setup which would avoid a single-point-of-failure.

A Source Code

The following python source code demonstrates how a program could iterrate over all proviced nameing services.

```
#!/usr/bin/env python
#
# python program which list all IOR contents.
# Several IORs and IORs with more than one object
#
# Stefan Stonjek
# 11-Feb-2004
#

# Standard/built-in modules.
import new, sys, os, re

# Fnorb modules.
from Fnorb.orb import CORBA, IIOP, IOP

# Naming Service modules.
from Fnorb.cos.naming import CosNaming

class NameServiceObj:

  def __init__(self,s_ior):
      self.maxRequestedNames = 100

      # initialise the ORB.
      orb = CORBA.ORB_init(sys.argv, CORBA.ORB_ID)

      # get the nameing server context
      self.nsContext = orb.string_to_object(s_ior)

      # get the entire naming service tree.
      self.getNamesInContext(self.nsContext, "")

  def resolveName(self, name, nsContext = None):
      returnObj = None
      if nsContext:
          returnObj = nsContext.resolve(name)
      else:
          returnObj = self.nsContext.resolve(name)
      return returnObj

  def getNamesInContext(self, nsContext, contextName, parentContextName = "/"):
      (bindingList, bindingIterator) = nsContext.list(self.maxRequestedNames)
      if (bindingIterator != None):
          while (1):
              (result, newBindingList) = bindingIterator.next_n(self.maxRequestedNames)
              if result:
                  bindingList = bindingList + newBindingList
              else:
                  break
      for b in bindingList:
          #for method in dir(b.__class__):
          #    print method
          if (b.binding_type == CosNaming.ncontext):
              newNsContext = self.resolveName(b.binding_name, nsContext)
              newContextName = contextName + "/" + b.binding_name[0].id
              newContext = self.getNamesInContext(newNsContext, newContextName, newContextName)
          else:
              full_context = parentContextName + ’/’ + b.binding_name[0].id
              print ’%-60s %-20s’ % (full_context , b.binding_name[0].kind)

def main(argv):

    env_vars= [’SAM_NAMING_SERVICE_IOR’,’SAM_NAMING_SERVICE_IOR_1’,’SAM_NAMING_SERVICE_IOR_2’,’SAM_NAMING_SERVICE_IOR_3’]

    for env_var in env_vars:
      sam_naming_service_ior = os.getenv(env_var)
      if sam_naming_service_ior == None :
        return
      if re.compile("IOR:").search(sam_naming_service_ior, 0):

        ior = new.instance(IOP.IOR, {})
        ior._fnorb_from_string(sam_naming_service_ior)

        for p in ior.profiles:
          host         = p.profile_data.host
          port         = p.profile_data.port
          key          = p.profile_data.object_key
          version      = IIOP.Version(chr(1), chr(0))
          type         = ior.type_id
          profile_body = IIOP.ProfileBody(version, host, port, key)
          profile      = IOP.TaggedProfile(IOP.TAG_INTERNET_IOP, profile_body)
          p_ior        = IOP.IOR(type, [profile])
          s_ior        = p_ior._fnorb_to_string()
          print "+++ " + host + " " + str(port) + " " + key + " " + type + " ++++++++"
          ns = NameServiceObj(s_ior)

    return

if __name__ == ’__main__’:
    sys.exit(main(sys.argv))
```